VNRVJIET

INDEX

Sl.No.	Expt. No.	Name of the Experiment	Date	Sheet No.	Remarks
		PART 1 : MICROPROCESSORS			
1.	1.	Programs for 16 bit arithmetic operations for 8086.		1-2	
	A.	Addition of two 8 bit no's	26th Dec, 18	3	
	B.	Addition of two 16 bit no's		4	
	C.	Subtraction of two 8 bit no's		5	
	D.	Subtraction of two 16 bit no's		6	3/4/19
	E.	Multiplication of two 8 bit no's		7	
	F.	Multiplication of two 16 bit no's		8	
	G.	Division of 16 bit no by 8 bit no.		9	
	H.	Division of two 16 bit no's		10	
	A.	Multiplication using IMUL	2nd Jan, 19	11	
	I	Division using IDIV		12	
	J.	Multibyte Addition		13-14	
	K.	Multibyte Subtraction		15-16	3/4/19
	L.	Checking for Overflow		17-18	
	M.	Invert lower nibble	9th Jan, 19	19	
	N.	Evaluate Arithmetic		20	
	O	expression.			

VNRVJIET

INDEX

Sl No.	Expt. No.	Name of the Experiment	Date	Sheet No.	Remarks
	P.	Fibonacci series	9th Jan, 19	21-22	
	Q.	Factorial of a no.		23	
	R.	LCM of two no's		24	
	S.	Average of n no's		25	
	T.	Reverse bits in a byte		26	
	U.	Swap the nibbles		27	
	V.	Count no. of 0's & 1's		28	3/4/19
	W.	Binary to Gray Conversion	23rd Jan, 19	29	
		ADDRESSING MODES.			
	1.	Addition using Index based addressing		30	
	2.	Base relative + index addressing mode		31-32	
2	2.	SORTING AN ARRAY FOR 8086			
	(i)	Ascending order		33-34	
	(ii)	Descending order		35-36	
	(iii)	Smallest no. in array	30th Jan, 19	37-38	
	(iv)	Largest no. in array		39-40	3/4/19
3	3.	Program for Searching a no. or character in a string for 8086.		41-42	

VNRVJIET

INDEX

Sl No.	Expt. No.	Name of the Experiment	Date	Sheet No.	Remarks
4.	4.	STRING MANIPULATIONS FOR 8086			
		(1) Moving a string	6th Feb,19	43-44	
		(2) Check for character in a string		45-46	
		(3) Concatenating two strings		47-48	
		(4) Reverse of a string		48-49	
		(5) Compare two strings		50-51	31/4/19
5	5.	Program for digital Clock design using 8086.	13th Feb,19	52-53	
6	6.	Interfacing ADC and DAC to 8086.		54-56	
		(1) ADC Interfacing	20th Feb,19.	57-58	
		(2) DAC Interfacing;		59-60	
		(i) Square wave gen.			
		(ii) Ramp wave gen.		61	31/4/19
7.	7.	Interfacing Stepper Motor to 8086	20th Feb,19.	62-63	

VNRVJIET

INDEX

Sl. No.	Expt. No.	Name of the Experiment	Date	Sheet No.	Remarks
		PART 2 : MICROCONTROLLERS			
		Steps for working with Keil µvision - 3		64 - 65	
8	8.	PROGRAM USING ARITHMETIC, LOGIC AND BIT MANIPULATION INSTRUCTIONS OF 8051.			
		(1) Performing arithmetic operations of 8051	6th Mar, 19	66	
		(2). Program for 16 bit addition		67	
		(3) Copying bytes in RAM		68	3/4/19
		(4) Transfering memory memory from ROM to RAM	13th Mar, 19	69	
		(5) Generation of square wave		70	
		(6) Sawtooth wave gen.		70	
		(7) Triangulare wave gen.		71	
9.	9.	Program and Verify Timer / Counter in 8051			
		(1) Square wave using timer 0 in mode 1	21st Mar, 19.	72	
		(2) Square wave using timer 1 in mode 2		73	

VNRVJIET

INDEX

Sl.No.	Expt. No.	Name of the Experiment	Date	Sheet No.	Remarks
10	10.	UART operation in 8051.			
		(1) Send a character	27th Mar, 19	74	
		(2) Send a word.		75	
		(3) Recieve a character		76	
		(4) Recieve a string		77.	
11	11.	Interfacing LCD to 8051.	30th Mar, 19.	78 – 80	3/4/19
12	12.	Interfacing Matrix Keyboard to 8051		81 – 83	

PART 1: MICROPROCESSORS AND INTERFACING

→ The softwares used for simulating 8086 are MASM (Microsoft assembler) and TASM (Turbo Assembler)

→ MASM: The microsoft macro assembler (MASM) is an x86 assembler that uses the intel syntax for MS-DOS and microsoft windows.

→ Steps for programming:

~~C:\users\admin>cd C:~~

1) Z:\>mount c c:\

Z:>C:

C:\>cd 8086

C:\8086>

} Change Z to C drive and directory to 8086

2) Editor: This is used to open the editor for entering code

C:\8086> edit filename.asm.

3) Compilation: This is used to verify the code for errors in masm

C:\8086> masm filename.asm.

4) Linking: This is used to link the asm file to the obj file.

C:\8086> link filename.obj

5) Execution and Debugging :-

The obj file is executed to generate the executable file

C:\8086> debug filename.exe

-g = this is used for entire program execution
-t = line by line execution
-u0 = unassemble the code
-q = to exit the debugger.
-d <address of segment>:0 [unassembled dumped value of segment]

Eg:- d 076b:0

Basic 8086 program structure.

Any 8086 program consists of two parts:

(i) Directives (for the assembler)
(ii) Instructions and values (for the compiler)

EXPT - I

PROGRAMS FOR 16 bit ARITHMETC OPERATIONS FOR 8086.

A) Addition of two 8-bit numbers:

Registers used : AL, BL

Software used : Micro assembler (MASM)

Program:

```
assume  cs: code      ; move code segment
code  segment         ; starting code
mov  al, 13h          ; moving 13h into AL register
mov  bl, 10h.         ; moving 10h into BL register
add  al, bl           ; adding al & bl contents
int  3h               ; break point interrupt
code  ends            ; end of code segment
end                   ; end of prom
```

CALCULATIONS:-

Adding two 8 bit no's

al = 13h = 0001 0011

bl = 10h = 0001 0000

al = 23h = 0010 0011

OUTPUT:-

AX = 0023 BX = 0010 CX = 0000 PX = 000

OV UP EI PL NZ AC PE CY.

B) Addition of two 16 bit numbers.

Registers used : AX, BX

Software used : Macro assembler (MASM)

Program:

```
assume cs:code      ; moving code segment
code segment        ; starting code
mov ax, 13h         ; moving 13h to AX register
mov bx, 10h         ; moving 10h to BX register
add ax, bx          ; adding contents of AX, BX
int 3h              ; break point interrupt
code ends           ; end of code segment
end.                ; end of program.
```

CALCULATIONS:-

Adding two 16 bit no's

ax = 0013h = 0000 0000 0001 0011

bx = 0010h = 0000 0000 0001 0000

ax = 0023h = 0000 0000 0010 0011

OUTPUT:-

AX = 0023 BX = 0010 CX = DX = 0000

OV UP EI PL NZ AC PE CY

9) Subtraction of two 8-bit numbers:

Registers used : AL, BL

Software used : Macro assembler (MASM)

Program:

```
assume cs: code      ; naming code segment
code segment         ; starting code
mov al, 59h          ; moving 59h to Al register
mov bl, 29h.         ; moving 29h to Bl register
sub al, bl           ; subtracting contents of Al, Bl.
int 3h               ; break point interrupts
code ends            ; end of code segment
end.                 ; end.
```

CALCULATIONS :-

Subtraction of two 8 bit no's

al = 59h	=	0101 1001.	= 89d.
bl = 29h.	=	0010 1001	= 41d.
al = 30h	=	0011 0000.	= 48d.

OUTPUT :-

AX = FF30 BL = 0029 CX = DX = 0000

D) Subtraction of two 16-bit numbers

Registers used : AX, BX

Software used : Macro assembler (MASM)

Program:

```
Assume  cs : code      ; naming code segment
code segment
mov  ax, 29abh         ; move 29abh to ax
mov  bx, 0182h         ; mov 0182h to bx
sub  ax, bx            ; subtract contents of ax, bx
int 3h                 ; break point
code ends
end.
```

CALCULATIONS:-

Subtraction of two 16 bit no's.

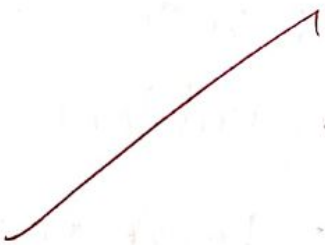
ax = 29abh = 0010 1001 1010 1011 = 10667d.

bx = 0182h = 0000 0011 0000 0010 = 386d

ax = 2829h = 0010 1000 0010 1001 = 10,281d

OUTPUT:-

AX = 2829 BX = 0182 CX = DX = 0000



E) Multiplication of two 8-bit no's.

Registers used : AL, BL

Software used : Macro assembler (MASM)

Program:

```
assume cs: code  ; name code segment
code segment
start:
        mov al, 9bh   ; move 9bh to al
        mov bl, 0c8h. ; move c8h to bl.
        mul bl.       ; multiply al & bl.
        code ends     ; end of code segment.
end start.
```

CALCULATIONS:

Multiplication of two 8 bits no's

al = 9bh = 1001 1011

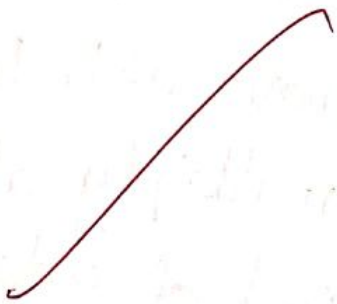
bl = c8h = 1100 1000

ax = 7918h = 0111 1001 0001 1000

OUTPUT :-

AX = 7918 BX = 00C8 CX = 0007

DX = 0000



F) Multiplication of two 16 bit no's

Registers used : AX, BX

Software used : Macro assembler (MASM)

Program:

```
assume cs: code  ; name code segment
code segment
start:
        mov ax, 56abh ; move 56abh to ax
        mov bx, 5a60h ; move 5a60h to bx
        mul bx ; multiply ax and bx
int 3h
code ends
end start.
```

CALCULATIONS:-

Multiplication of two 16 bit no's

ax = 56abh = 0101 0110 1010 1011.

bx = 5a60h = 0101 0010 0110 0000

ax = 9E20 = 1001. 1110. 0010. 0000

dx = 1E98 = 0001. 1110 1001 1000

Product : 1E98 : 9E20

OUTPUT:-

AX = 9E20 BX = 5A60 CX = 0009 DX = 1E98.

9) Division of 16 bit no. by 8 bit no:

Registers used : ax, bl.

Software used : Macro assembler (MASM)

Program:

```
assume  cs : code
code segment
start :
        mov  ax, 089bh ; move 89bh to ax
        mov  bl, 05ch.  ; mov 5ch to bl.
        div  bl.  ; 16 bit | 8bit div.
int 3h
code ends
end start.
```

3|4|19.

CALCULATIONS :-

Division of 16 bit by 8 bit no.

ax = 089bh. = $(2203)_{10}$.

bl = 05ch. = $(92)_{10}$.

ax/bl = 92 ⟌2203 (= Quotient = 23

Remainder = 87

Quotient = al = 17h.

Remainder = ah = 57h.

OUTPUT :-

AX = 5717 BX = 005C CX = 0008 DX = 0000

4) Division of two 16 bit no's.

Registers used : dx, ax bx.

Software used : Macro assembler (MASM)

Program:

```
assume cs: code      ; naming code segment.
code segment
start:
        mov dx, 0    ; clearing dx
        mov ax, 0007h ; move 7h to ax
        mov bx, 0003h ; move 3h to bx
        div bx       ; perform ax/bx.
        int 3h
code ends
end start.
```

CALCULATIONS:-

Division of two 16 bit no's.

ax = 0007h = dividend

bx = 0003h = divisor.

$\frac{ax}{bx}$ ⇒ Quotient = 0002h

Remainder = 0003h.

OUTPUT:-

Ax = 0002 Bx = 0003 Cx = 000C.

Dx = 0001.

NV UP EI PL NZ NA PO NC

I) Multiplication of two 8 bit no's using Imul.

Registers used: al, bl.

Software used: Macro assembler (MASM)

Program:

```
assume cs: code
Code segment
start:
        mov al, 02h     ; move 2h into al
        mov bl, 0fdh    ; move -3h = 0fdh into bl.
        imul bl         ; signed multiplication al * bl.
        int 3h
Code ends
end start.
```

CALCULATIONS :-

al = 02h = 0000 0010.

bl = FDh = 1111 1101 = -03h.

X

ax = FFFAh = 1111 1111 1111 1010

OUTPUT :-

AX = FFFA BX = 00FD CX = 0007 DX = 0000

7) Division of two 16 bit no's using IDIV :-

Registers used : dx, ax, bx

Software used : Macro assembler

Program

```
assume cs: code
code segment
start:
        mov dx, 0      ; clear dx
        mov ax, 0fffah ; move fffah to ax
        mov bx, 0002 h ; move 2h to bx
        idiv bx        ; signed div. ax/bx
        int 3h

code ends
end start
```

CALCULATIONS :-

ax = FFFAh = -6d

bx = 0002h. = 2d.

$$2\overline{)-6}\,(-3$$
$$\underline{-6}$$
$$0$$

Quotient = -3 = FDh

Remainder = 0 = 00h.

OUTPUT :-

AX = 00FD

AX → Quotient

remainder

BX = 0002 CX = 0008.

DX = 0000.

Name of the Laboratory: MPMC Lab

Experiment No. 1K Date. 2/1/19

8) MULTI BYTE Addition:

Registers used : AX, DS, SI, DI, BX, CX, AL

Software used : Macro assembler (MASM)

Program:

```
assume cs: code , ds: data.
data segment                ; data segment starts
num1 db 12h, 34h, 60h, 11h  ; giving multiple
num2 db 4ch, 01h, 29h, 0ACh ;  data.
res db 7 dup(?)  ; storing for result.
data ends.

code segment
start : mov ax, data
        mov ds, ax
        mov SI , offset num1.  ; num1 addr in SI
        mov DI, offset num2  ; num2 addr in DI
        mov BX, offset res.  ; res addr. in BX
        ADD SI, 3
        ADD DI, 3
        ADD BX, 3
        CLC
```

3/4/19.

OUTPUT :-

-g

AX = 075E BX = 0007 CX = DX = 0000

SP = BP = SI = DI = 0000

DS = 076A ES = 075A SS = 0769 CS = 0768

-eds : 0000 { give any no. } ↵

076A : 0000 12.34.60.11 4C.01.29.AC.

n1 n2

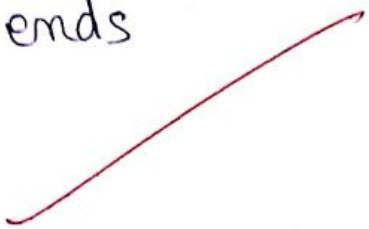
I/P ref. location ↳ press tab untill we get all I/Ps.

Press Tab

076A : 0008 5E 35 89 BD 00 00 00 00

addition of n1 & n2 bytes

```
X:  mov AL, [SI]   ; move value at SI to AL
    mov DL, [DI]   ; move value at DI to DL
    SUB AL, DL     ;
    mov [BX], AL
    INC SI
    INC DI
    INC BX
    LOOP X

int 3h
code ends
end.
```



2) Multibyte Subtraction :-

Registers used : AX, DS, CX, SI, DI, BX, AL, DL

Software used : Macro assembler [MASM]

Program:

```
assume cs:code , ds:data

data segment
num1 db 04h, 05h, 04h ; define multiple
num2 db 03h, 02h, 01h. ; bytes to perform
res db 00h, 00h, 00h. ; multibyte sub
count db 03h
data ends

code segment
org 1000h ; start of program at 1000h.
mov ax, data ; moving data to ds
mov ds, ax ; through ax
mov cx, count
mov si, offset num1 ; Storing addresses of
mov di, offset num2 ; num1, num2, res in
mov Bx, offset res. ; si, di and bx
```

OUTPUT :- In command window

-g

AX = 0709 BX = 0009 CX = DX = 0000

NV UP EI PL NZ NA PE NC.

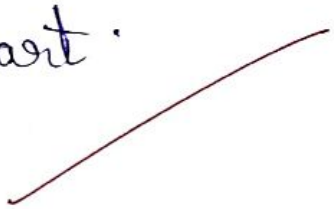
-eds : 0000 ↵

076A : 0000 04 05 0A → n1 bytes

03 02 01 → n2 bytes

076A : 0008 01 03 09 → Multibyte subtraction

```
        MOV CX,4
back :  MOV AL, [SI]
        ADC AL, [DI]
        MOV [BX], AL
        DEC SI
        DEC DI
        DEC BX
        Loop back

int 3h
Code ends
end start.
```



Name of the Laboratory: MPMC Lab

Experiment No. 9M Date. 9/1/19

M) Write an ALP to print "OFFSET" or "OFF NOT SET" if any overflow in addition of two no's.

Registers used: AX, DS, SI,

Software used: MASM.

Program:

```
assume cs: code, ds: data.
data segment
 s1 db -80h.
 s2 db -18h
 str1 db "OFF SET"
 str2 db "OFF NOT SET"
data ends

code segment.
start: mov ax, data
       mov ds, ax
       mov si, offset s1
       mov di, offset s2
       mov ax, [di]
       add ax, [si]
```

```
        jo set
        mov dx, offset str2
        mov ah, 09h.
        int 21h
        jmp final.
set :   mov dx, offset str1
        mov ah, 09h
        int 21h.
final : int 03h
        code ends
        end start.
```

3/4/19.

OUTPUT :- In command window

(i) -g ↵

of not set (no overflow)

AX = 0968 BX = 0000 CX = 0046 DX = 0008

NV UP EI PL NZ NA PO CY

(ii) If I/P is S1 ⇒ 80h , S2 ⇒ -a0h

-g ↵

of ~~not~~ set (overflow)

AX = 0986 BX = 0000 CX = 0046 DX = 0008

OV UP EI PL NZ NA PO CY

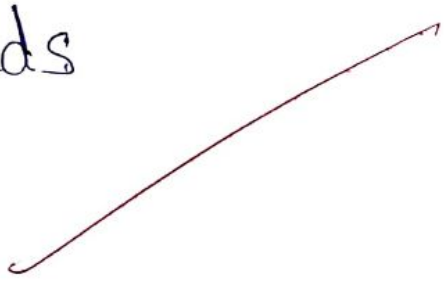
N) Program to invert the lower nibble of 45h.

Registers used : Ax, Bx

Software used : MASM.

Program :

```
assume  cs:code
code segment
mov  ax, 45h        ; move value to be inverted
 XOR  Ax, 0Fh.      ; XOR the value
 MOV  BX, AX        ; store inverted value in bx
 Code ends
 End.
```



OUTPUT :- Invert lower nibble.

45 hex → [4] [5]

[4] ↓ higher nibble

[5] ↳ lower nibble.

binary form : 0100 0101

Invert lower nibble : 0100 1010 ⇒ [4A]

AX = 004A BX = 004A CX = 0009 DX = 0000

NV UP EI PL NZ NA PO NC

0) Program to evaluate an arithmetic expression $3x^3 - 2x + 8$

Registers used: AX, BX, DX

Software used: MASM

Program:

```
assume cs:code
code segment
start : mov ax, 02h
        mov bx, 02h
        mul bx
        mul bx
        mov bx, 03h
        mul bx
        mov dx, ax
        mov ax, 02h
        mov cx, 02h.
        mul cx
        mov bx, ax
        sub dx , bx
        mov ax , dx
        add ax, 08h
int 3h
code ends
end start
```

CALCULATION :-

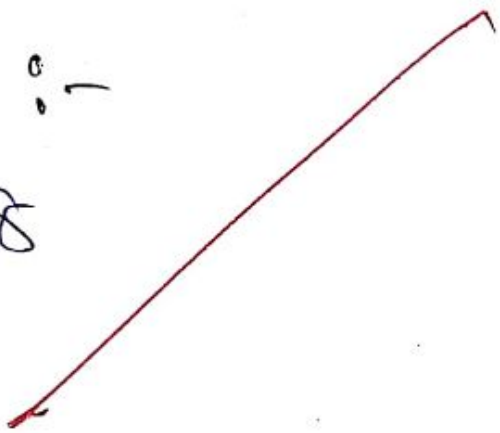
Expression : $3x^3 - 2x + 8$

$x = 2$

$3(2)^3 - 2(2) + 8 = 28$

OUTPUT :-

AX = 28



P) To find and print the Fibonacci series

Software used : MASM.

Program:

```
Assume cs:code, ds:data.
data segment
res db 10 dup(10h)
data ends

code segment
start: mov ax, data.
       mov ds, ax
       mov di, offset res
       mov al, 00h
       mov bl, 01h.
       mov [di], al.
       mov res, di
       inc di
       mov [di], bl.
       mov res, di
       inc di
       mov ch, 08h .
```

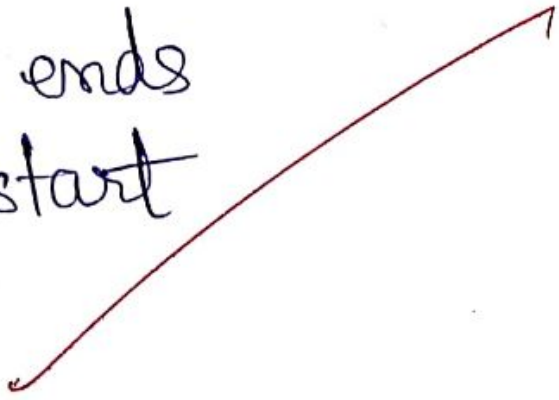
VNRVJIET

Name of the Laboratory: MPMC Lab

Name of the Experin: Matti Fibonac

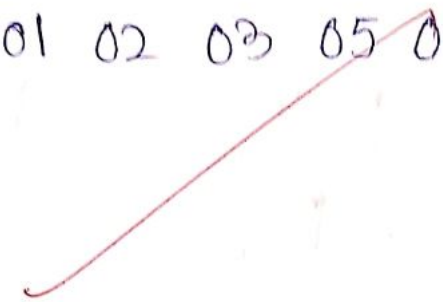
Experiment No. 12

```
l1: mov cl, al.
    add al, bl.
    DAA
    mov [di], al.
    mov res, di
    mov bl, cl.
    inc di
    dec ch.
    jnz l1

int 03h
code ends
end start
```



OUTPUT :- Fibonacci series

076A:0000 00 01 01 02 03 05 08 - 13 21 00 00

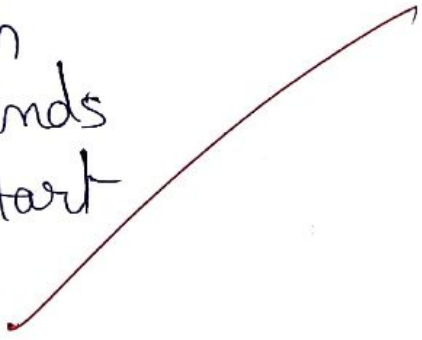


Q) To find the factorial of a given number.

Software used: Macro assembler (MASM)

Program:

```
assume cs: code
code segment
start : mov si, 0000h
        mov cl, [si]
        mov al, 01h
x:  MUL cl.
    DEC cl.
    JNZ x
    int 03h
    code ends
    end start
```



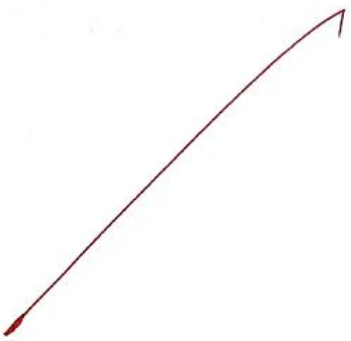
CALCULATION (Factorial) :

If no = 5

then 5! = 1x2x3x4x5 = 120. = 78h

OUTPUT :-

AX = 0078



R) To find the LCM of two no's

Registers used: Ax, Bx, Cx, Dx

Software used: MASM.

Program:

```
assume cs: code
code segment
start : mov ax, 03h
mov bx, 11h
back : mov dx, 0
mov cx, bx
div bx
mov ax, bx
mov bx, dx
cmp bx, 0
jne back
go : mov ax, 3
mov bx, 11
mul bx
div cx
int 03h
code ends
end start
```

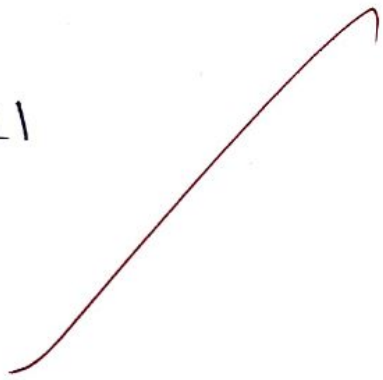
CALCULATION: (LCM of two no's)

If n1 = 3 and n2 = 11.

then LCM (3, 11) = 33 d = 21 h.

OUTPUT:

AX = 0021



Name of the Laboratory: MPMC lab. | Experiment No. 15 Date. 9/1/19

5) Find the average of 'n' numbers

Registers used : ax, bx, cx

Software used : MASM.

Program:

```
assume cs:code  ; name segments
code segment    ; start of code
start : mov cx, 03h ; move 3h into cx
        mov bx, cx  ; mov cx into bx
        mov ax, 00h ; clear ax

x : add ax, cx  ; add ax and cx
    loop x
    div bx      ; divide ax by bx
    int 03h.
code ends
end start
```

CALCULATION:- (Avg of n no's)

If $n = 3$,

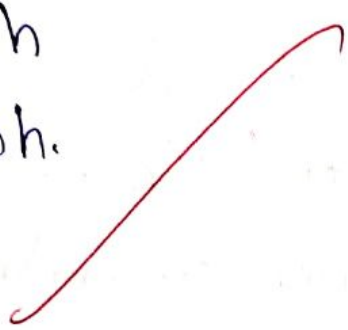
then avg of first 'n' no's $= \frac{1+2+3}{3} = 2$

OUTPUT :-

AX = 0002h

BX = 0003h

CX = 0000h.



NRVJIET

ame of the Laboratory: MPMC lab

Name of the Experiment: Reverse bits in a byte.

Experiment No. 14 Date. 23/1/19

T) Reverse the bits in a byte.

Registers used : AX, BX, CX

Software used : MASM.

Program :

```
assume cs:code  ; naming segments
code segment
start : mov ax,10101100b ; move 10101100b into ax
mov bx, 0h          ; clear Bx
mov cx, 08h.        ; move 8h into Cx
l1 : rcr ax, 1      ;rotate Ax right by 1
rcl bx, 1           ;rotate Bx left by 1
loop l1
mov ax, bx
int 03h
code ends
end start
```

CALCULATION :-

The given I/P no' is

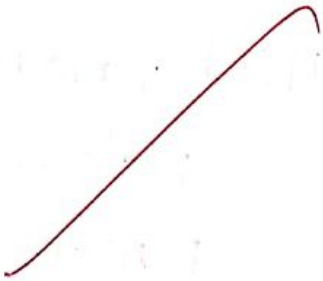
1010 1100 = AC h.

Reversing the bits we get = 0101 0011

3 5h.

OUTPUT :-

AX = 0035h.



v) Swap the nibbles

Registers used: AL, CL

Software used: MASM.

Program:

```
assume cs, code
code segment
mov al, 84h.    ; move 84h into AL
mov cl, 4       ; move 04h into CL
rot al, cl      ; rotate AL left by 4
int 03h
code ends
end.
```

CALCULATIONS:- (Swap nibbles)

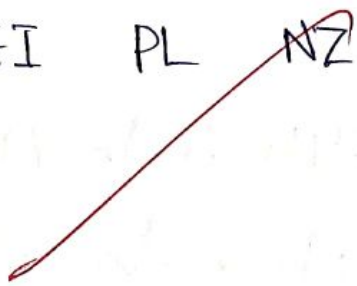
84 (hex) ⇒ upper nibble = 8
lower nibble = 4

Swapping both = 48h.

OUTPUT:-

AX = FF48 BX = 0000 CX = 0004 DX = 0000

NV UP EI PL NZ NA PO NC.



V) Count the no of 0's and 1's in a given no.

Registers used : AX, BX, CX, DX

Software used : MASM.

Program :

```
assume cs: code
code segment
mov ax, 84h   ; move 84h into Ax
mov bx, 08h   ; move 8h into Bx
mov cx, 00h   ; clear cx
mov dx, 00h   ; clear Dx
jump2 : rcl al, 01h ; Rotate AL through carry by 1
jnc jump1   ; if carry ≠ 0, jmp to jump 1
inc cx
jmp jump3
jump1 : inc dx
jump3 ; dec bx
jnz jump2   ; jump to jmp 2 if ZF ≠ 0
int 03h
code ends
end.
```

CALCULATIONS :- (Zero's and Ones).

The given I/p no is : 84h

= 1000 0100

no. of zeros = 6

no. of ones = 2.

OUTPUT :-

CX = 0002

DX = 0006

W) Binary to Gray conversion

Registers used:- AX, BX

Software used: MASM.

Program:

```
assume cs: code
code segment
mov ax, 1234h ; move 1234h into AX
mov bx, ax    ; move ax to bx
shr bx, 01h   ; shift right bx by 01h
xor ax, bx    ; XOR of AX & BX
int 03h
code ends
end.
```

3/4/19.

CALCULATION :-

Given binary no is 1234 h.

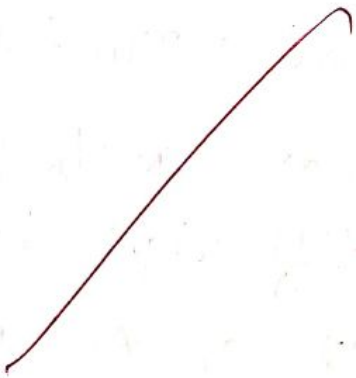
= (0001 0010 0011 0100)$_2$

= (0001 1001 0010 0110) ⇒ eqt gray code.

= 1326 h.

OUTPUT :-

AX = 1326 h



ADDRESSING MODES

1) Program to perform addition using Index based addressing modes.

```
assume cs: code, ds: data
data segment
arr db 10h, 20h, 30h, 40h, 50h.
data ends

code segment
mov ax, data
mov si, offset arr
mov bx, 3h
mov ax, bx
mov bx, [si+bx]
add ax, bx
code ends
end.
```

CALCULATION:—

AX = 3h + [si + 3]

= 3h + 30h

= 33h.

OUTPUT :—

—g

AX = 0033

2) Program for base relative + Index addressing mode:

Registers used: AX, DS, SI, AL, CL

Software: MASM.

Program:

```
ECE1 segment
roll 1  db  17h, 45h, 22h, 52h, 25h, 55h, 29h, 57h.
roll 2  db  13h, 60h, 15h, 50h, 18h, 45h, 20h, 52h.
roll 3  db  19h, 60h, 28h, 49h, 25h, 52h, 13h, 58h.
student equ roll3
intm equ 0
extm equ 1.
subj equ 2
ece1 ends.

assume cs:code, ds:ece1
code segment
start: mov ax, ece1
       mov ds, ax
       mov bx, offset student
```

```
mov al, subj
dec al.
mov cl, 2
mul cl
mov si, ax
mov ch, [bx+si+intm]
int 3h
code ends
end start.
```

3/4/19.

CALCULATION:-

Load bx = roll 3 offset add

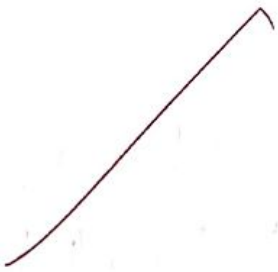
ch = [roll 3 + si + 2]

= 60h.

OUTPUT:-

CX = 6000

CH.



EXPT - 2

SORTING AN ARRAY FOR 8086.

1. Program to sort an array in ascending order

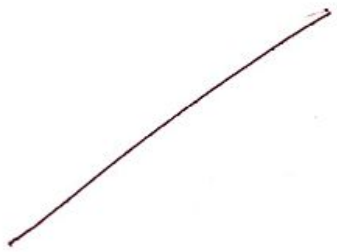
Registers used : AX, DS, SI, CX, DX

Software used : Macro assembler (MASM)

Program:

```
data segment    ; start data segment
arr1 db 49h, 64h, 23h, 12h, 92h. ; give array values
data ends       ; end data segment

code segment
assume cs:code, ds:data ; naming segments
start : mov ax, data
        mov ds, ax
        mov dx, 4 ; no of rounds
back2 : mov cx, 4 ; no of comparisions
        mov si, offset arr1. ; load arr1 address
back1 : mov al, [si]    ; mov [si] to a.l.
        cmp al, [si+1] ; compare values
```

```
        jb next1    ; jump if below to next1.
        xchg al, [si+1] ; exchange corresponding
        xchg al, [si]   ;  array values.
next1:  inc si
        loop back1.  ; repeat the loop
        dec dx.
        jnz back2.   ; jump if not zero to back2
int 3h
code ends
end start.
```



OUTPUT :-

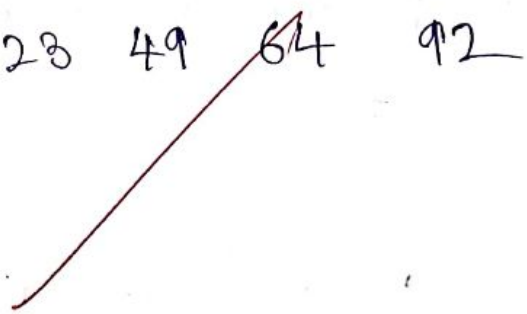
-g

AX = 0719 BX = 0000 = CX = DX

DS = 076A

-d DS:0000

076A:0000 12 23 49 64 92



2. ALP to sort an array in descending order

Registers used: AX, CX, CX, SI, DS.

Software used: Macro assembler (MASM)

Program

```
data segment        ; start data segment
array db 05h, 25h, 91h, 26h.  ; giving array values
data ends.          ; end data segment

code segment
assume cs:code, ds:data.  ; naming segments
start : mov ax, data  ; move data to ax
        mov ds, ax    ; move ax to ds
        mov dx, 03h   ; move 03h to dx
back1 : mov cx, dx    ; move dx to cx
        LEA si, array ; load effective address
back2 : mov al, [si]
        cmp al, [si+1]
        ja next       ; jump above next
        xchg [si+1], al.  ; exchange the
        xchg [si], al.    ; resp. values
```

```
next : inc si      ; increment si
       dec cx      ; decrement cx
       loop back2  ; go back2 (repeat loop)
       dec dx
       jnz back1 ; jump if not zero.
int 3h     ; break point interrupt
Code ends  ; end of code segment
end start. ; end of program.
```

OUTPUT :-

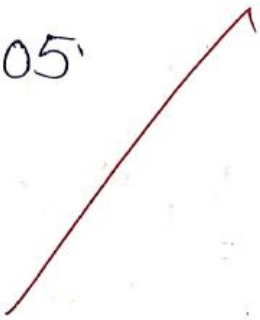
-g

AX = 0791 BX = 0000 = CX = DX

SI = DI = 0000 DS = 076A.

-d DS : 0000.

076A : 0000 91 26 25 05



3. ALP to find the smallest no. in an array.

Registers used: AX, DS, SI, CX, DI

Software used: Macro assembler (MASM)

Program:

```
data segment
array db 50h, 20h, 40h, 60h, 70h
smaller db 01h dup(?)
data ends

assume cs: code, ds: data
code segment
start: mov ax, data;
       mov ds, ax
       mov cx, 04h.
       LEA si, array
       LEA di, smaller
       mov al, [si]
back: cmp al, [si+1];
      jb next;
```

```
        mov al, [si+1];
next :  inc si;
        dec cx;
        jnz back;
        mov [di], al;

int 3h
code ends
end start.
```

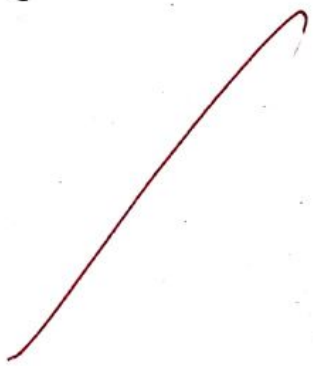
3/4/19.

OUTPUT :-

-d DS:0000

076A:0000 50 20 40 60 70 20

AX = 20 (Smallest no)



4. ALP to find the largest no. in an array

Registers used: AX, ST, DS, CX, DI.

Software used: Macro assembler (MASM)

Program:

```
assume cs: code, ds: data
 data segment
 array db 52h, 24h, 0A2h, 07C7h, 27h;
 res db 01h dup (0)
data ends.

code segment
start : mov dx, data
        mov ds, ax
        LEA si, array
        LEA di, res
        mov cx, 0004h
        mov al, [si]
```

```
back :  cmp al, [si]
        ja next
        mov al, [si+1]

next :  inc si
        loop back

int 03h
code ends
end start.
```

Scanned by CamScanner

OUTPUT :

AX = 07C7 (Largest no.)

EXPT-3

PROGRAM FOR SEARCHING A NUMBER OR CHARACTER IN A STRING FOR 8086.

Registers used: AX, DS, SI, CX, AL.

Software used: Macro assembler (MASM)

Write an ALP to seach for the character 'j' in a given string "vnrvjiet" and display "character is found" if it has "j" else print "character not found".

```
data segment
str1 db "vnrvjiet$"
res1 db "character is found $".
res2 db "character not found $"
data ends.

code segment
assume cs:code, ds:data.
start:
        mov dx, data
        mov ds, ax
```

```
        mov si, offset str1
        mov cx, 8
        mov al, "j"
back :  cmp al, [si]
        je last1
        mov dx, offset res1
        mov ah, 09h
        int 21h
        jmp last2

last1 : mov dx, offset res2
        mov ah, 09h
        int 21h

last2 : int 3h

Code ends
end start
```

OUTPUT :-

Character is found.

EXPT - 4.

STRING MANIPULATIONS FOR 8086.

1. Program to move a source string to destination (copy the string).

Registers used: ax, ds, es, si, di, cx
Software used: Macro assembler (MASM).

Program:

```
data segment
srcstr db "vnrvjiet $"
len dw ($ - srcstr);
data ends.

extra segment
desstr db 10 dup(?)
extra ends

Code segment
assume cs:code, ds: data, es: extra.
```

```
start:  mov  ax, data
        mov  ds, ax
        mov  ax, extra
        mov  es, ax
        cld
        mov  si, offset srcstr
        mov  di, offset desstr
        mov  cx, len
        rep  movsb
        int  3
code ends
end start.
```

OUTPUT :-

i) -u0 (gives DS, ES addresses)

ii) -g (executes program)

iii) -d 076a : 0

076A:0000 76 6E 72 ... 74-24 vnrvjet$

ASCII values

iv) -d : 076b : 0

076B:0000 76 6E 72 ... 74 -24 vnrvjet$

2. Program to check for "j" in a given string. using string instructions

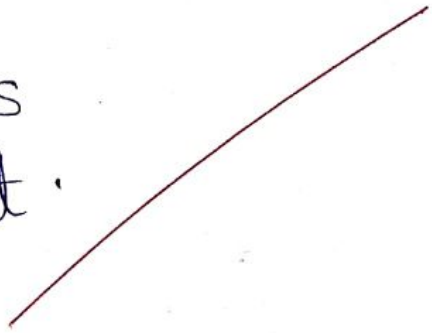
Registers used: Ax, Cx, DS, SI, ES.

Software used: Macro assembler (MASM)

Program:

```
data segment
desstr1 db "vnrvjiet$"
res1 db "character j is found"
res2 db "character j is not found"
data ends.

code segment
assume cs:code, ds:data, es:data
start: mov ax, data
       mov ds, ax
       mov es, ax
       cld
        mov di, offset desstr1
       mov cx, 8
       mov al, "j"
```

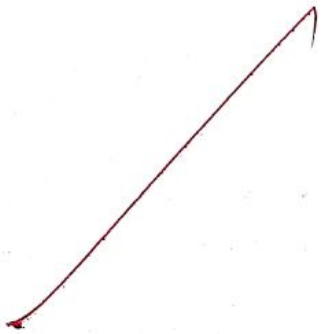
```
repne scasb
je last1
mov dx, offset res2
mov ah, 09h.
int 21h
jmp last2

last1 : mov dx, offset res1
        mov ah, 09h
        int 21h

last 2 : int 3

code ends
end start.
```



OUTPUT:-

Character j is found

AX = 096A BX = 0000 CX = 0003 DX = 0008



3. Program to concatenate two strings

Registers used: AX, DS, ES, SI, DI, CX

Software used: Macro assembler (MASM)

Program:

```
data segment
srcstr1 db "Akash$"
len1 dw ($ - srcstr1)
srcstr2 db "Moses$"
len2 dw ($ - srcstr2)
data ends.

extra segment
desstr db 15 dup(?)
extra ends

code segment
assume cs: code, ds: data, es: extra.
start: mov ax, data
       mov ds, ax
       mov ax, extra
       mov es, ax
       cld.
       mov si, offset srcstr1
       mov di, offset desstr.
       mov cx, len1.
```

```
        dec cx
        rep movsb
        mov si, offset srcstr2
        mov cx, len2
        rep movsb
        int 3
code ends
end start.
```

4. Program to reverse a string.

Registers used: AX, CX, ES, DI, DS, SI.

Software used: Macro assembler (MASM)

Program:

```
data segment
srcstr db "vmrvjiet $"
len1 dw ($ - srcstr1)
data ends

extra segment
desstr db 15 dup(?)
extra ends.
```

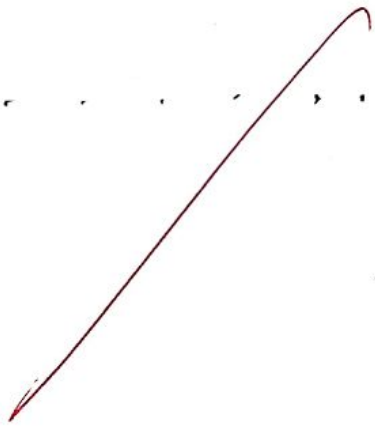
OUTPUT :- (Concatenate strings)

i) -u0

ii) -g

iii) -d 076b:0

076b:0000 . . . . . akash moses $.

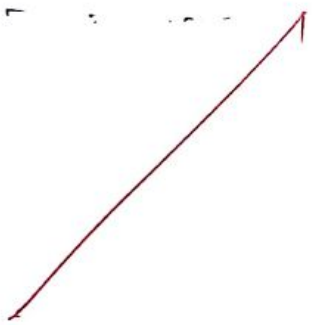


```
code segment
assume cs: code, ds: data, es: extra
start : mov ax, data
        mov ds, ax
        mov ax, extra
        mov es, ax
        mov si, offset srcstr1
        mov di, offset desstr
        mov cx, len1
        dec cx
        add di, cx
        move byte ptr es:[di], $
        dec di
back :  cld
        load sb.
        std
        sto sb.
        loop back
        int 3
code ends
end start
```

OUTPUT (Reverse a string)

-u
-g
-d 076b:0000

076b:0000 . . . . . t eijrnv $



5. Program to compare two strings :

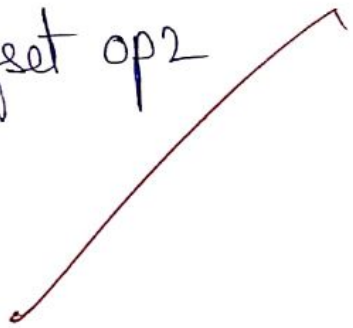
Registers used : AX, CX, DS, SI, ES, DI

Software used : Macro assembler (MASM)

Program :

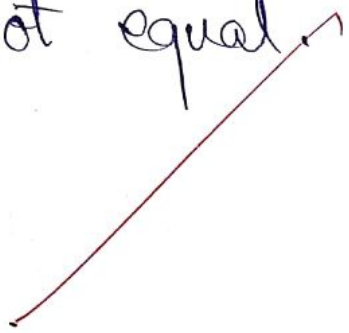
```
data segment
str1 db " vnrvjiet 1"
str2 db " vnrvjiet 2"
op1 db " Strings are equal $"
op2 db " Strings are not equal $".
count equ 9
data ends

code segment
assume cs: code , ds: data
start : mov ax , data.
        mov ds, ax
        mov ax, data
        mov es, ax
        mov cx, count.
```

```
        mov si, offset str1
        lea di, str2
        cld
        repe cmpsb
        jnz next
        mov ah, 09h
        mov dx, offset op1.
        int 21h
        jmp last

next :  mov ah, 09h
        mov dx, offset op2
        int 21h.

last :  int 3h.

code ends
end start.
```



OUTPUT:-

Strings are not equal.



EXPT-5.

PROGRAM FOR DIGITAL CLOCK DESIGN USING 8086.

Registers used: AX, DS, CX

Software used: MASM.

Program:

```
assume cs: code , ds: data.

data segment
year dw 1 dup (?)
moth dw 1 dup (?)
day db 1 dup (?)
day of week db 1 dup (?)
hours db 1 dup (?)
mins db 1 dup (?)
sec db 1 dup (?)
data ends.
```

```
Code segment
assume cs: code , ds: data.
start : mov ax, data
        mov ds, ax
        mov ah, 2Ah
        int 21h
        mov year, cx
        mov month, dh
        mov day, dl.
        mov day of week, al.
        mov ah, 2ch
        int 21h
        mov hours, ch
        mov min, cl
        mov sec, dh
        int 3h
        Code ends

end start
```

OUTPUT :-

E2 07 03 15 03 0F 0B → time (min)

year → E207
month → 03
date → 15
day → 03 [Wednesday]
time (hr) → 0F

EXPT - 6

INTERFACING ADC/DAC to 8086.

→ In this experiment we interface bot ADC and DAC to 8086.

→ The basic steps which are to be followed for any interfacing experiment of 8086 are:

Steps for Execution:

→ C:\masm123> masm filename ↵

→ C:\masm123> link filename ↵

→ C:\masm> promview ↵

Promview software:

(Promview software is used for Binary to hex file conversion)

i) File operations: (F).

Select Format ↵

press 0 - Binary.(EXE) ↵

(ii) File operations: ↵

Read ↵

Filename.EXE: ↵

default 0000

↵

(iii) RAM operation ↵

View ↵

Starting address : org. address + 200

= 4200

ending address : 4300

Press ESC key.

(iv) File operation: ↵

Select Format ↵

Press 1 → intel hex ↵

(v) File operation: ↵

write ↵

RAM starting address : 4200

RAM ending address : 4230 ↵

Enter Filename : ADC.Hex ↵

→ Exit - Back to DOS

Are you sure (Y/N) - Y

Are you saving file (Y/N) - Y

→ C:\masm123> edit filename.hex

Post Promview steps :

→ Erase first line in hex editor file

→ C:\masm123> teserial 1

press 'S' in kit keyboard

VIK-86 (configuration)

type - 'L' & OFFSET - 0 ↵

Enter filename {= adc.hex ↵

Running the program:

→ The program can be run using two Commands

(i) D4200 : Unassembles & displays linewise code

(ii) G4200 : It is usede to run the program.

ADC Interfacing :

→ In interfacing an ADC to the 8086 µP in lab, we have a fixed set of port addresses:

PA : FF50H PC : FF54H

PB : FF52H Control word reg : FF56H.

→ We use port A as I/P port and Port C upper as SOC signal.

Registers used : Dx, AL, ex

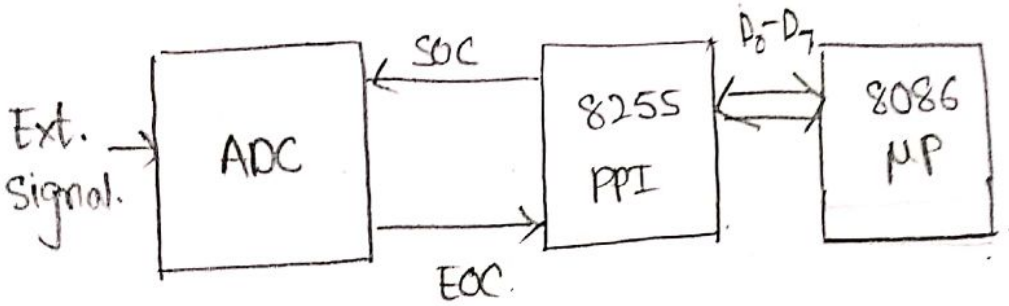
Software used : MASM, Promview.

Program :

```
code segment
assume cs: code
org 4000h ; starting of prg for promview
start :
        mov dx, OFF56h; control word addr reg.
        mov al, 90h ; storing control word in CWR.
        out dx, al ; sending control word.
        mov dx, OFF54h; port C reg (I/P)
```

```
mov al, 0FFh    ; sending
out dx, al      ; active
mov al, 00h     ; low SOC
out dx, al.     ; signal
mov al, 0FFh    ; in this part
out dx, al.     ; of code
call delay      ; delay generation
call delay.     ; for EOC.
mov dx, 0FF50h; port A as I/P port.
in al, dx
int 3h

delay: mov cx, 0FFFFh ; assuming max value
       loop1 : nop
               nop.
       dec cx
       jnz loop1
       ret
code ends
end start.
```

ADC Interfacing :-



SOC : on PC_0, short low duration 

EOC : maximum delay generation. (nop).

OUTPUT :-

- D4200 ↵ (unassembles and displays hex file)
- G4200 ↵ (Runs the prg & displays digital data)

I/P voltage (potentiometer)	O/P digital value
0	00
1	34
2	69
3	A0
4	D2
5	FF

DAC Interfacing :-

→ Here we generate various waveforms with digital data with almost all ports configured as o/p ports.

1) Program to generate square waveform using a DAC.

Registers used : AL, DX , CX.

Software : MASM , Promview

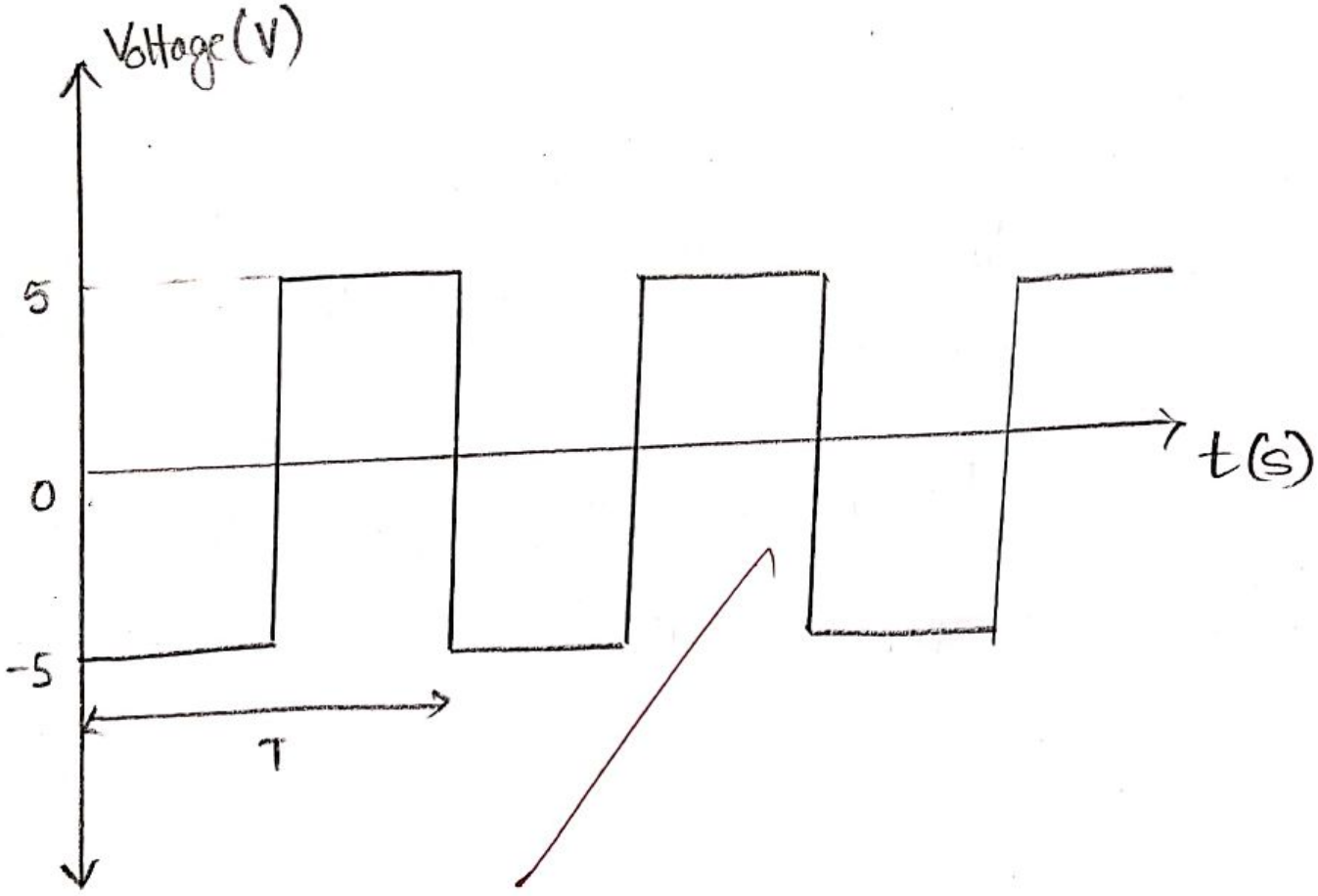
Program :

```
Code segment
assume  CS: code.

org 2000h ; starting address
mov dx, 0FF56h ; control word reg.
mov al, 80h. ; control word value
out dx, al
mov dx, 0FF52h ; using portB
next :
      mov al, 0FFh.
```

```
        out dx, al.
        call x
        mov al, 00h
        out dx , al
        call x
        mov al,
jmp next
 int 3h
x : mov cx, 0FFF1h

loop 1 : nop
         nop
         dec cx
         jnz loop 1

ret
code ends
end start.
```

OUTPUT :-

Square wave generated in CRO.



2) Program to generate Ramp waveform in a DAC using 8086.

Registers used: AL, DX, CX

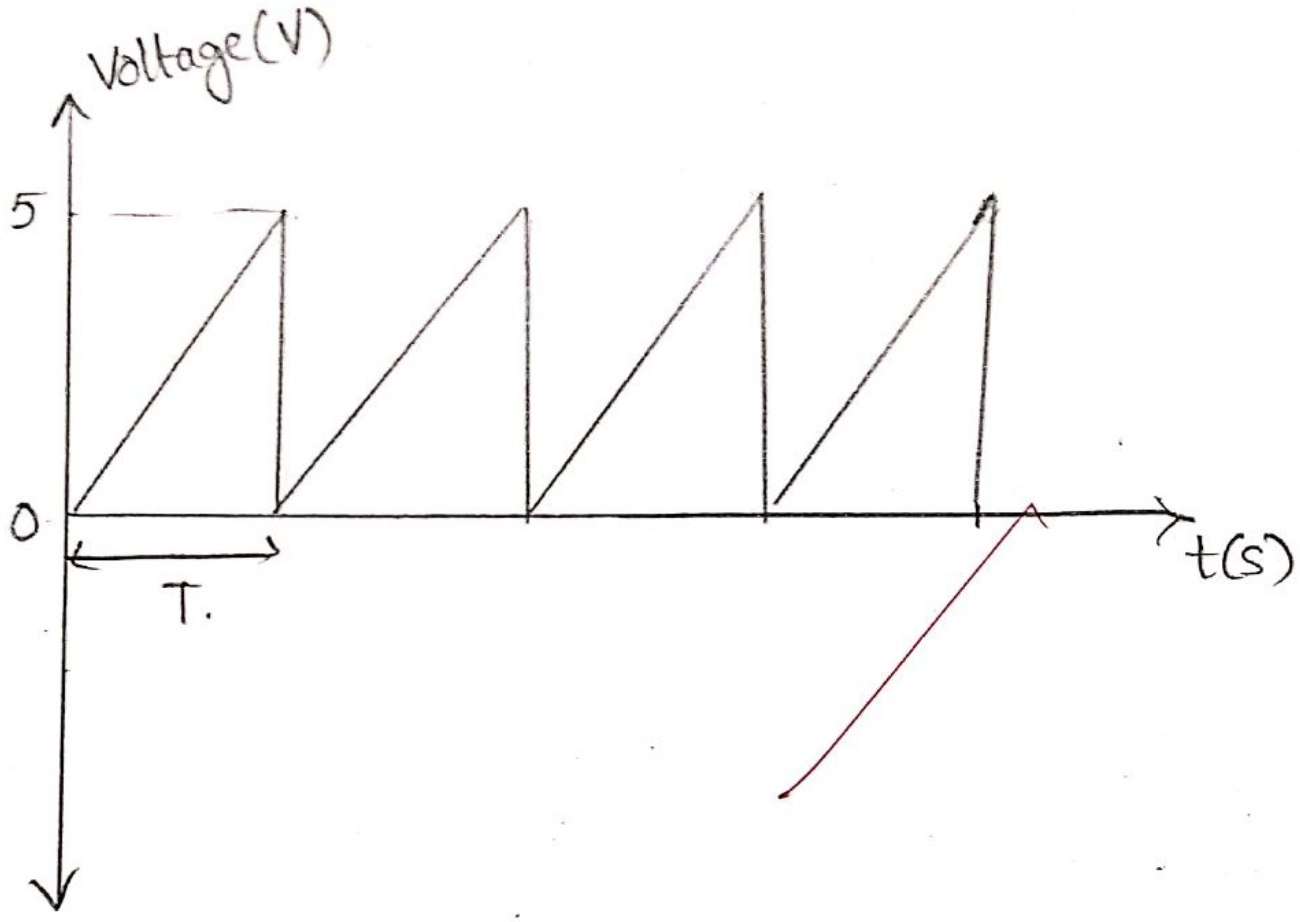
Software: MASM, Promview

Program:

```
code segment
assume cs: code.
org 2000h
mov dx, 0FF56 h.
mov al, 80h
out dx, al.
mov dx, 0FF52h
mov al, 00h
next : out dx, al
       inc al
       jmp next
int 3h
code ends
end start
```

OUTPUT:-

Ramp wave generated in CRO.



EXPT - 7

INTERFACING STEPPER MOTOR TO 8086.

→ The stepper device is used to produce stepwise rotation in motor. We also use the similar promview steps as in ADC/DAC in stepper motor interfacing.

Program:

Registers used : AL, DX, BX

Software : MASM, Promview

```
mov dx, FF26    ; CWR
mov al, 80      ; enabling CWR reg
out dx, al      ; sending control word
step : mov dx, FF20  ; port A to Px
       mov al, 80    ; 1st seq. data
       out dx, al.
       call delay    ; call delay subroutine.
       mov al, A0    ; 2nd data seq.
       out dx, al.
       call delay.
```

```
mov al, E0      ; 3rd sequence of data.
out dx, al.
call delay.

mov al, C0      ; 4th data sequence.
out dx, al.
call delay
jmp step.       ; to repeat stepping process

delay: mov bx, 0010 ; decides rotation speed.
last : mov al, FF.

last1 : nop
        nop
        nop
        nop
        dec al
        jnz last1
        dec bx
        jnz last

ret.
```

3141 19.

Stepper motor:-

Step angle = 1.8° per step

Switching sequence = 4 steps

Sequence for one step clockwise rotation

	PA_7	PA_6	PA_5	000 0	
Step 1:	1	0	0	"	= 80
Step 2:	1	0	1	"	= A0
Step 3:	1	1	1	"	= E0
Step 4:	1	1	0	"	= C0

Port addresses:

PA = FF20 PC = FF24

PB = FF22 Control Reg = FF26

Output:

The stepper motor interfaced rotates in steps (1.8° per step.)